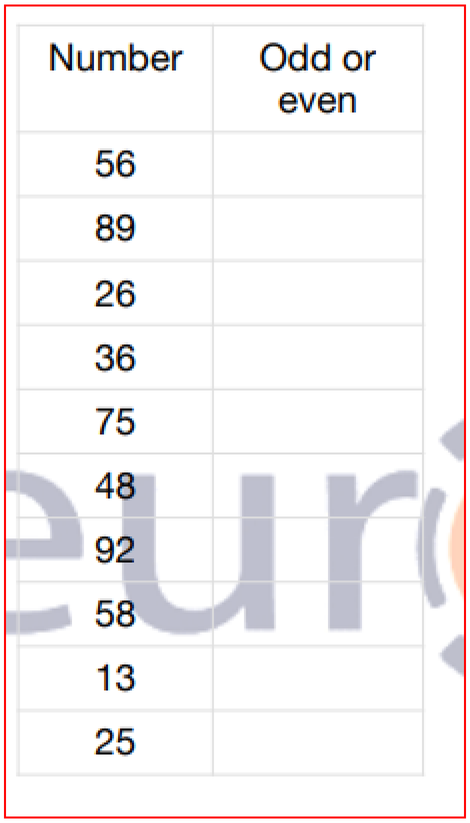
Excel Assignment – 20

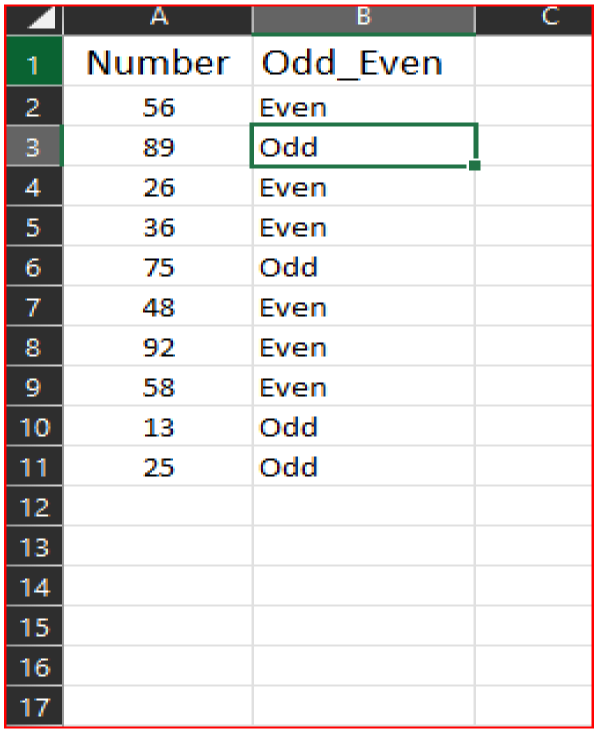
1. **Write a VBA code to select the cells from A5 to C10. Give the name "Data Analytics" and fill the cells with the following cells "This is Excel VBA"**

Number Odd or even

56 89 26 36 75 48 92 58 13 25

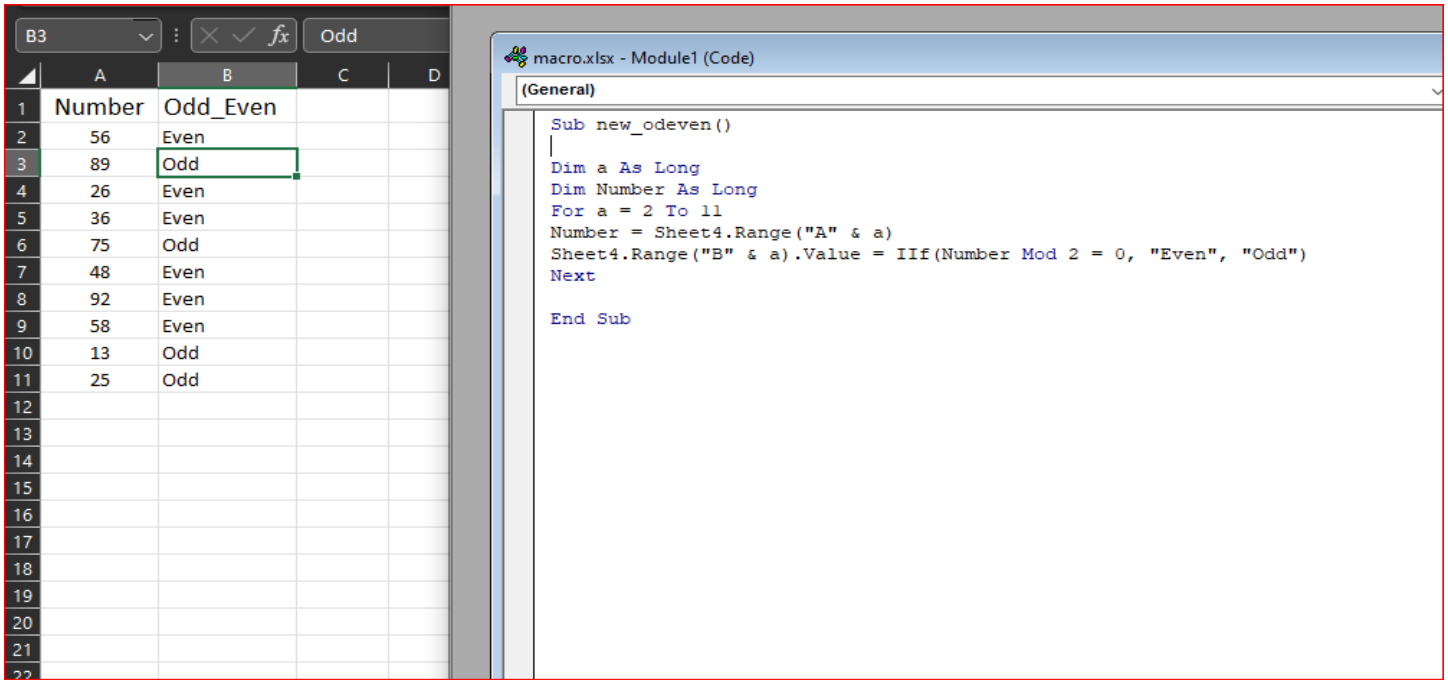


Number	Odd or even
56	
89	
26	
36	
75	
48	
92	
58	
13	
25	



	A	B	C
1	Number	Odd_Even	
2	56	Even	
3	89	Odd	
4	26	Even	
5	36	Even	
6	75	Odd	
7	48	Even	
8	92	Even	
9	58	Even	
10	13	Odd	
11	25	Odd	
12			
13			
14			
15			
16			
17			

2. **Use the above data and write a VBA code using the following statements to display if the number is odd or even in the next column. IF ELSE statement b. Select Case statement c. For Next Statement**



	A	B
1	Number	Odd_Even
2	56	Even
3	89	Odd
4	26	Even
5	36	Even
6	75	Odd
7	48	Even
8	92	Even
9	58	Even
10	13	Odd
11	25	Odd

```
Sub new_odeven()

Dim a As Long
Dim Number As Long
For a = 2 To 11
Number = Sheet4.Range("A" & a)
Sheet4.Range("B" & a).Value = IIf(Number Mod 2 = 0, "Even", "Odd")
Next

End Sub
```

3. What types of errors do you usually see in VBA?

➢ In Visual Basic, errors fall into one of three categories: syntax errors, run-time errors, and logic errors.

- Syntax errors are those that appear while writing code. If we're using Visual Studio, Visual Basic checks code by typing it in the Code Editor window and alerts us if making a mistake, such as misspelling a word or using a language element improperly. If compile from the command line, Visual Basic displays a compiler error with information about the syntax error.

- Run-time errors are those that appear only after you compile and run your code. These involve code that may appear to be correct in that it has no syntax errors, but that will not execute. For example, you might correctly write a line of code to open a file. But if the file does not exist, the application cannot open the file, and it throws an exception.

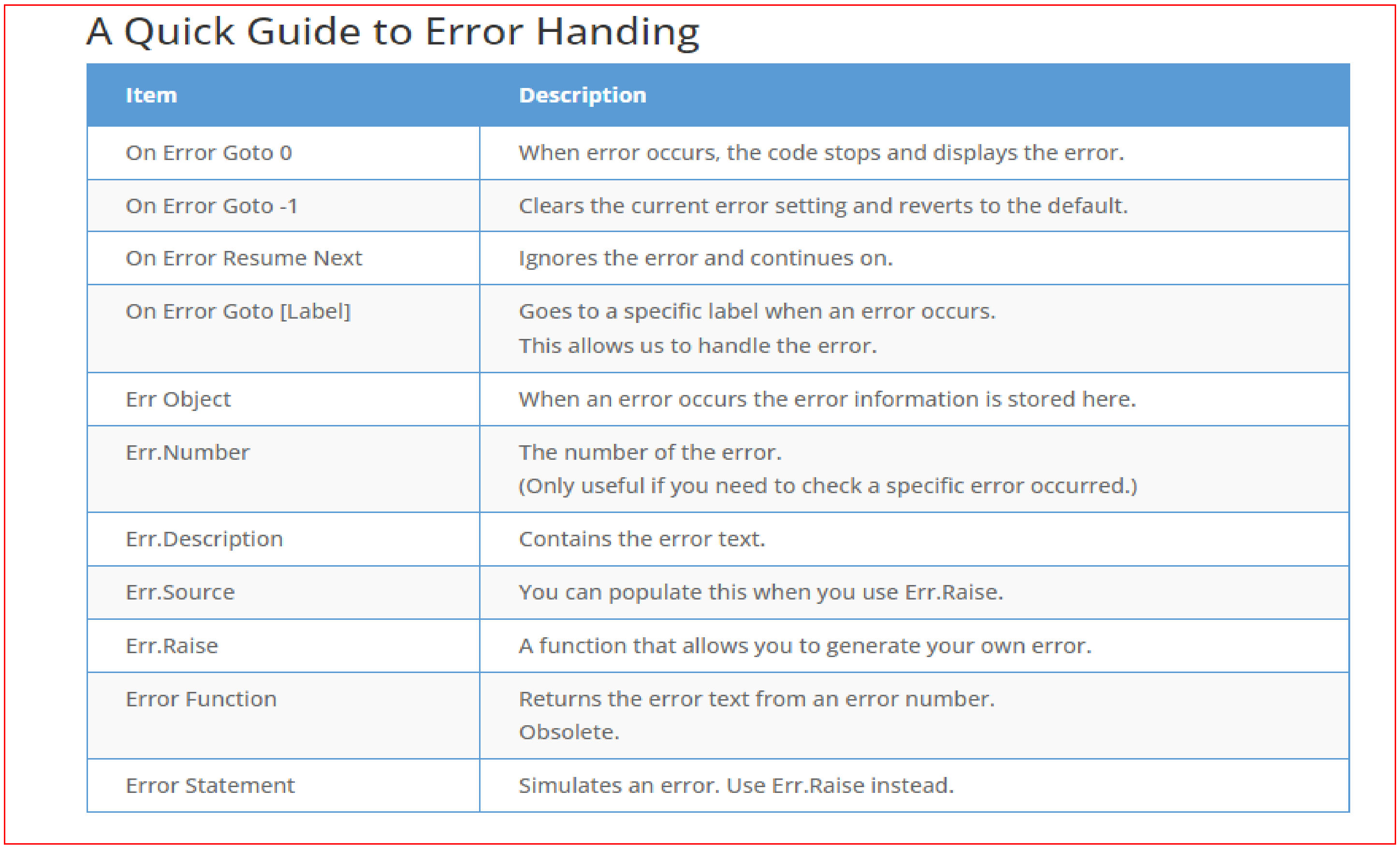
- Logic errors are those that appear once the application is in use. They are most often faulty assumptions made by the developer or unwanted or unexpected results in response to user actions. For example, a mistyped key might provide incorrect information to a method, or you may assume that a valid value is always supplied to a method when that is not the case.

4. How do you handle Runtime errors in VBA?

- We can fix most run-time errors by rewriting the faulty code or by using exception handling, and then recompiling and rerunning it.
- To handle an error inline, use the Resume Next statement with On Error. Any errors that occur during runtime cause Info Connect to continue executing the macro at the next statement. If an error occurs, it is handled by opening a dialog box and passing control to another procedure or to a routine within the same procedure.

5. Write some good practices to be followed by VBA users for handling errors.

- Use 'On Error Go [Label]' at the beginning of the code. ...
- Use 'On Error Resume Next' ONLY when you're sure about the errors that can occur.
- When using error handlers, make sure you're using Exit Sub before the handlers. Use multiple error handlers to trap different kinds of errors.

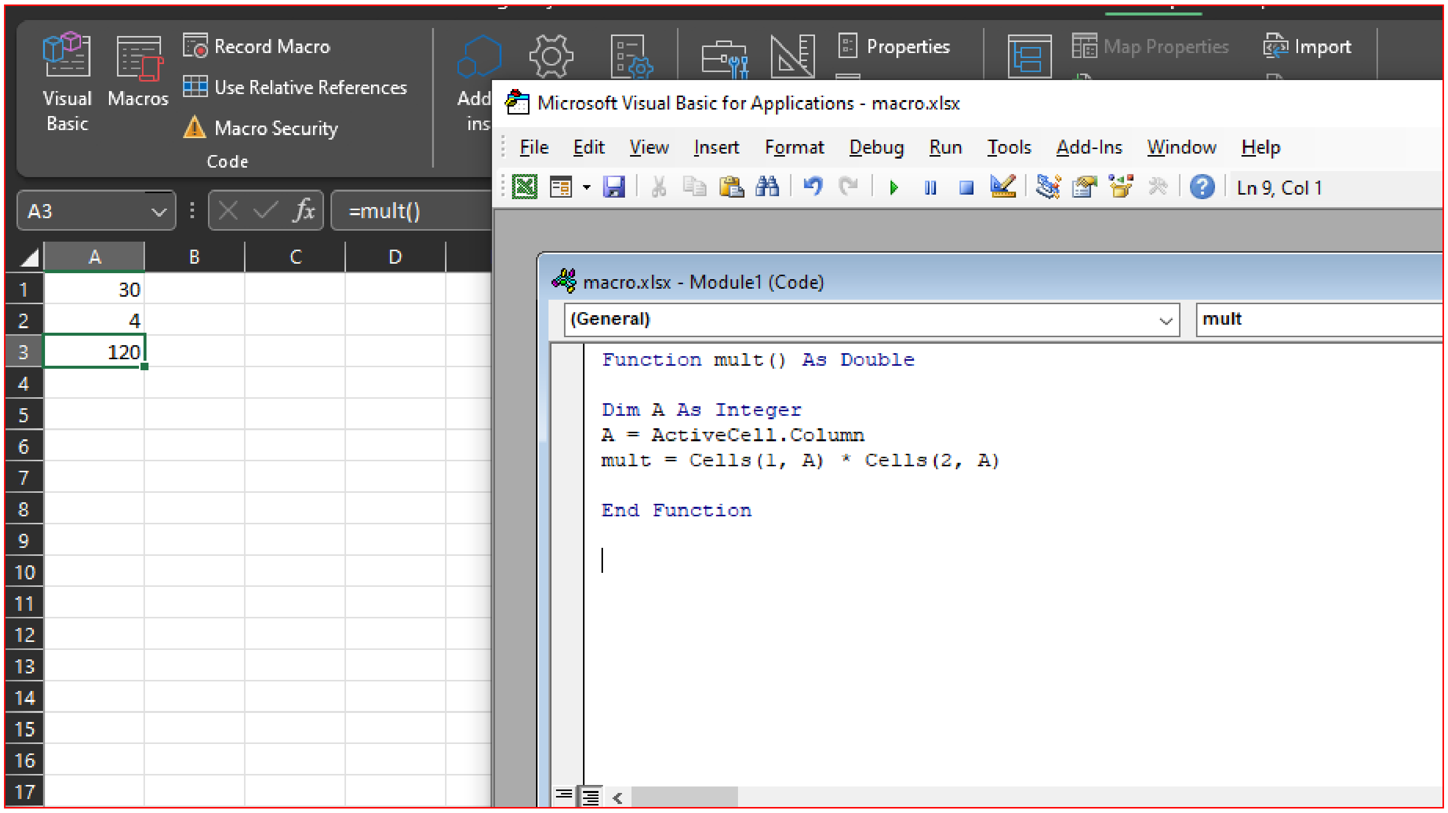


A Quick Guide to Error Handing

Item	Description
On Error Goto 0	When error occurs, the code stops and displays the error.
On Error Goto -1	Clears the current error setting and reverts to the default.
On Error Resume Next	Ignores the error and continues on.
On Error Goto [Label]	Goes to a specific label when an error occurs. This allows us to handle the error.
Err Object	When an error occurs the error information is stored here.
Err.Number	The number of the error. (Only useful if you need to check a specific error occurred.)
Err.Description	Contains the error text.
Err.Source	You can populate this when you use Err.Raise.
Err.Raise	A function that allows you to generate your own error.
Error Function	Returns the error text from an error number. Obsolete.
Error Statement	Simulates an error. Use Err.Raise instead.

6. What is UDF? Why are UDFs used? Create a UDF to multiply 2 numbers in VBA.

➢ We can create our functions using VBA coding, which is technically called "User-Defined Functions" (UDF). They are also called "custom functions" in Excel VBA. Any formula we can access from the worksheet with a piece of code is called UDF.


Visual Basic
Macros
Record Macro
Use Relative References
Macro Security
Code
Properties
Map Properties
Import
Microsoft Visual Basic for Applications - macro.xlsx
File Edit View Insert Format Debug Run Tools Add-Ins Window Help
Ln 9, Col 1
A3
=mult()
macro.xlsx - Module1 (Code)
(General)
mult

	A	B	C	D
1	30			
2	4			
3	120			

```
Function mult() As Double

Dim A As Integer
A = ActiveCell.Column
mult = Cells(1, A) * Cells(2, A)

End Function
```